(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 May 2001 (03.05.2001)

PCT

(10) International Publication Number
WO 01/31794 A1

(51) International Patent Classification[7]: H03M 7/42, H04N 7/50

(21) International Application Number: PCT/EP00/10715

(22) International Filing Date: 26 October 2000 (26.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
99402712.6 29 October 1999 (29.10.1999) EP

(71) Applicant *(for all designated States except US)*: KONINKLIJKE PHILIPS ELECTRONICS N.V. [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: PENAIN, Stéphane [FR/NL]; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

(74) Agent: DEN BRABER, Gerard; Internationaal Octrooibureau B.V., Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

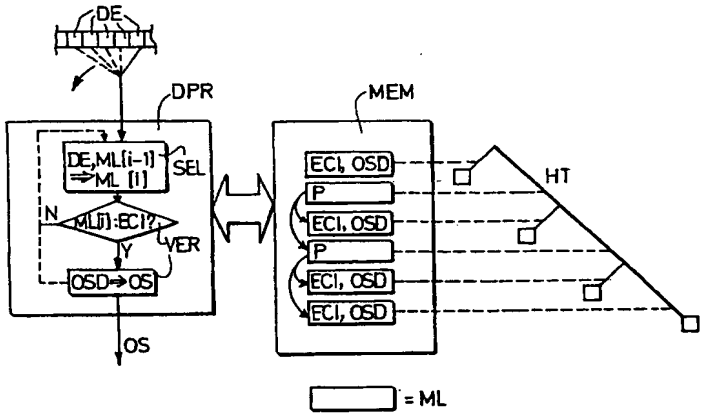
(81) Designated States *(national)*: CN, JP, KR, US.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: VARIABLE-LENGTH DECODING



(57) Abstract: Data which has been variable-length encoded in accordance with a Huffman tree (HT), is decoded in the following manner. A memory (MEM) is used in which the Huffman tree has been stored so that branches of the Huffman tree are represented by memory locations (ML). If a memory location represents a branch that is followed by subsequent branches, the memory location contains a pointer (P) to memory locations representing these subsequent branches. If the memory location represents a branch that is not followed by subsequent branches, the memory location contains an end-of-code indication (ECI) and output-symbol data (OSD). A decoding processor (DPR) carries out a selection step (SEL) and an evaluation step (EVA) with each data element (DE) in a sequence of data elements. In the selection step, the decoding processor selects a memory location on the basis of the data element and the content of the most recently selected memory location. In the evaluation step, the decoding processor checks if the memory location that has been selected contains the end-of-code indication. If so, the decoding processor provides an output symbol (OS) on the basis of the output-symbol data. This manner of variable-length decoding allows cost-efficient implementations, in particular when various types of variable-length codes need to be decoded. It is particularly suitable for use in an MPEG decoder.

WO 01/31794 A1

Variable-length decoding.

FIELD OF THE INVENTION

The invention relates to decoding data which has been variable-length encoded in accordance with a Huffman tree. The invention may be applied, for example, in an MPEG decoder (MPEG is an abbreviation of Motion Pictures Expert Group).

BACKGROUND ART

Variable-length encoding is based on the following principle. Data to be encoded is composed of various different symbols, just like a text is composed of various different words. A symbol which frequently occurs in the data is coded by means of a relatively short code word, that is, a code word which comprises relatively few bits. A symbol which rarely occurs in the data is coded by means of a longer code word. For example, let it be assumed that data is composed of four symbols: A, B, C and D. Let it further be assumed that the data is composed of symbol A for 50 %, of symbol B for 25% and of symbols C and D for 12.5 % each. The symbol A may be represented by a 1-bit binary code word, the symbol B by a 2-bit binary code word, and the symbols C and D by a 3-bit binary code word. When the data is encoded with these code words, the average code-word length in the encoded data will be less than 2 bits.

A basic aspect of variable-length encoding is how to distinguish between different code words. If the code words are fixed in length, that is if each code word has a length of 2 bits for example, we know that there will be a new code word every 2 bits. If code words are not fixed in length, which is the case in variable-length encoding, another technique should be applied to distinguish between different code words.

Variable-length encoding in accordance with a Huffman tree allows to distinguish between different code words in an efficient manner. What follows is an example of a Huffman tree for encoding the four symbols A, B, C and D presented hereinbefore. A first-level node constitutes the beginning of the tree. There are two first-level branches connected to this node. One first-level branch has a leaf at its end which leaf represents the symbol A. A binary zero (0) is associated to this branch. The other first-level branch has a second-level node at its end. A binary one (1) is associated to this branch. There are two

CONFIRMATION COPY

second-level branches connected to the second-level node. One second-level branch has a leaf at its end, which leaf represents the symbol B. A binary zero (0) is associated to this branch. The other second-level branch has a third-level node at its end. A binary one (1) is associated to this second-level branch. There are two third-level branches connected to the third-level node. Each third-level branch has a leaf at its end, one leaf represents the symbol C and the other the symbol D. A binary zero (0) is associated to the branch whose leaf represents the symbol C. A binary one (1) is associated to the branch whose leaf represents the symbol D.

The Huffman tree described above provides the following code words. To reach the leaf representing symbol A, one must pass via the first-level branch to which a binary zero (0) is associated. The code word for symbol A is therefore 0. To reach the leaf representing symbol B, one must pass via the first-level branch to which a binary one (1) is associated and then via the second-level branch to which a binary (0) is associated. The code word for symbol B is therefore 10. To reach the leaf representing symbol C, one must pass via the first-level branch to which a binary one (1) is associated, then via the second-level branch to which a binary (1) is associated and then via the third-level branch to which a binary (0) is associated. The code word for symbol C is therefore 110. To reach the leaf representing symbol D, one must pass via the first-level branch to which a binary one (1) is associated, then via the second-level branch to which a binary (1) is associated and then via the third-level branch to which a binary (1) is associated. The code word for symbol D is therefore 111.

Data which has been variable-length encoded in accordance with a Huffman tree may be decoded in the following manner. A data register selects a sequence of bits from the data and supplies this sequence to a decoding logic circuit. The decoding logic circuit recognizes, as it were, a code word contained in the sequence of bits. Accordingly, the decoding logic circuit selects from the memory the symbol that represents the code word. For example, let it be assumed that data has been encoded with the code words 0, 10, 110 and 111 representing symbols A, B, C and D, respectively, as described hereinbefore. In that case, the decoding logic reads symbol A, B, C or D from the memory if the data register contains --0,-10, 110 or 111, respectively, the sign - representing a do-not-care value. A method of decoding as described in this paragraph appears to be disclosed in US Patent 5.828,907.

SUMMARY OF THE INVENTION

It is an object of the invention to allow cost-effective implementations of decoding data that has been encoded in accordance with a Huffman tree.

The invention takes the following aspects into consideration. The data compression factor that can be achieved by means of variable-length encoding basically depends on two factors. It depends on the statistical properties of the data to be encoded, and on the Huffman tree in accordance with which the data is encoded. A Huffman tree that provides a satisfactory compression factor for a given type of data, may not give a satisfactory compression factor for another type of data, because the two types of data have different statistical properties. For this reason, it is desirable to have different variable-length codes for different types of data, each code having its own specific Huffman tree.

In many applications, such as multi-media for example, it is desired to handle different types of data which have been variable-length encoded in different manners. In principle, this can be achieved by providing for each variable-length code, a dedicated variable-length decoder such as described hereinbefore in the background-art part. Such a solution requires a dedicated decoding logic circuit for each code. The decoding logic circuits will generally be relatively complex, because variable-length codes are generally relatively complex. Consequently, this solution requires an appreciable amount of circuitry and is therefore relatively expensive. Another solution is to use a general purpose processor and to provide variable-length decoding software for each variable-length code. However, this software-based solution will be relatively slow in terms of decoding throughput. It may require a fast general purpose processor in order to achieve a satisfactory decoding speed. Such a processor is relatively expensive and, therefore, the software-based solution will also be expensive.

In accordance with the invention, data which has been variable-length encoded in accordance with a Huffman tree, is decoded in the following manner. A memory is used in which the Huffman tree has been stored so that branches of the Huffman tree are represented by memory locations. If a memory location represents a branch that is followed by subsequent branches, the memory location contains a pointer to memory locations representing these subsequent branches. If the memory location represents a branch that is not followed by subsequent branches, which means the branch has a leaf at its end, the memory location contains an end-of-code indication and output-symbol data. A decoding processor carries out a selection step and an evaluation step with each data element in a sequence of data elements. In the selection step, the decoding processor selects a memory location on the basis of the data

element and the content of the most recently selected memory location. In the evaluation step, the decoding processor checks if the memory location that has been selected contains the end-of code indication. If so, the decoding processor provides an output symbol on the basis of the output-symbol data.

The decoding processor can be relatively simple, because it carries out a relatively simple standard procedure with each data element in the data to be decoded. What is more, the standard procedure does not depend on the Huffman tree. That is, the decoding processor can be used for carrying out different types of variable-length decoding. In order to allow different types of variable-length decoding, it is thus sufficient to replace one Huffman tree stored in the memory by another Huffman tree or to store different Huffman trees in the memory and to select a Huffman tree by defining a start address.

The invention thus allows decoding of various variable-length codes with relatively simple decoding circuitry. In contrast, the method of variable-length decoding described in the background art portion requires relatively complicated decoding circuitry in order to achieve the same. Variable-length decoding in accordance with the invention may require somewhat more memory than variable-length decoding described in the background art portion. However, for many applications, the invention will provide a saving on decoding circuitry, which will outweigh cost of additional storage capacity, if there is any such cost. Thus, the invention allows cost-effective implementations.

The invention and additional features, which may be optionally used to implement the invention to advantage, are apparent from and will be elucidated with reference to the drawings described hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a conceptual diagram illustrating basic features of the invention as claimed in claim 1.

Fig. 2 is a diagram illustrating a Huffman tree.

Fig. 3 is a block diagram illustrating a decoder for decoding data which has been encoded in accordance with the Huffman tree illustrated in Fig. 2.

Fig. 4 is a flow chart illustrating a method in accordance with which the decoder illustrated in Fig. 3 may operate.

Fig. 5 is a block diagram illustrating an MPEG decoder in accordance with the invention.

DETAILED DESCRIPTION OF THE DRAWINGS

The following remarks relate to reference signs. Like entities are designated by like letter references in all the Figures. Several similar entities may appear in a single Figure. In that case, a digit or a suffix is added to the letter reference in order to distinguish like entities. The digit or the suffix may be omitted for convenience or it may be replaced by an asterisk in the case where its value is not important (do not care value). This applies to the description as well as the claims.

Fig. 1 illustrate basic features for decoding data which has been variable-length encoded in accordance with a Huffman tree HT. A memory MEM is used in which the Huffman tree HT has been stored so that branches of the Huffman tree are represented by memory locations ML. If a memory location represents a branch that is followed by subsequent branches, the memory location contains a pointer P to memory locations representing these subsequent branches. If the memory location represents a branch that is not followed by subsequent branches, the memory location contains an end-of-code indication ECI and output-symbol data OSD. A decoding processor DPR carries out a selection step SEL and an evaluation step EVA with each data element DE in a sequence of data elements. In the selection step, the decoding processor selects a memory location on the basis of the data element and the content of the most recently selected memory location. In the evaluation step, the decoding processor checks if the memory location that has been selected contains the end-of-code indication. If so, the decoding processor provides an output symbol OS on the basis of the output-symbol data.

Fig. 2 illustrates an example of a Huffman tree. The Huffman tree defines a variable-length coding of four symbols: A, B, C and D. The Huffman tree comprises nodes NO, branches BR and leaves LF. There is a first, second and third-level node NO1, NO2 and NO3, respectively. There are two first, second and third-level branches BR1*, BR2*, and BR3*, respectively. A binary value, which may be zero (0) or one (1), is assigned to each branch BR. Leaves LF1, LF2, LF3 and LF4 represent symbols A, B, C and D, respectively.

The Huffman tree defines a code word for each symbol A, B, C and D in the following manner. The first-level node NO1 is taken as a starting node. From this node there is a certain path to each symbol. The code word is formed by the binary values assigned to the branches BR in the path in order of decreasing significance. For example, the path to symbol C is formed by the branches BR12, BR22 and BR31 to which the binary values 1, 1 and 0 are assigned, respectively. Consequently, the code word for symbol C is 110.

Fig. 3 illustrates a decoder for decoding data which has been encoded in accordance with the Huffman tree illustrated in Fig. 2. The decoder comprises a decoding processor DPR and a memory MEM. The memory comprises several memory cells @. Each memory cell @ has a specific address. The memory cells @0, @1 and @2 are divided in two halves. These memory cells comprise a left-half portion and a right-half portion. Each portion contains an address. It may further contain a flag √. For example, the left-half portion of the memory cell @0 comprises the flag √ and the address of memory cell @10. The memory cells @10, @11, @12 and @13 comprise the symbols A, B, C and D, respectively.

The memory MEM of the decoder illustrated in Fig. 3 contains the Huffman tree illustrated in Fig. 2. The Huffman tree has been stored in a particular manner. The memory cells @0, @1 and @2 represent the first, second and third-level branches, BR1*, BR2* and BR3*, respectively. The left-half portions of these memory cells represent the branches BR*1 which have been assigned a binary zero (0). The right-half portions represent the branches BR*2 which have been assigned a binary one (1). If a portion of a memory cell @ represents a branch BR which is connected to subsequent branches, the portion contains the address of the memory cell which represents these further branches. For example, the right-half portion of the memory cell @0 represents branch BR12 of the Huffman tree. This branch is further connected to the branches BR21 and BR22. The memory cell @1 represents these branches. Therefore, the right-half portion of the memory cell @0 contains the address of memory cell @1. If a portion of a memory cell @ represents a branch BR which has a leaf LF at its end, the portion contains the flag √. The portion further contains the address of the memory cell which contains the symbol that belongs to this leaf.

Fig. 4 illustrates a method in accordance with which the decoder illustrated in Fig. 3 may operate. The method comprises six steps S1-S6. Steps S1-S4 are carried out with each subsequent bit of the data to be decoded. Steps S5 and S6 are carried out only when the current bit forms the last bit of a code word. In that case, the decoder provides an output symbol.

In step S1, the decoding processor reads a memory cell @ which has been selected as the next-to-be-read memory cell during a series of steps S1-S6 carried out previously: RD[NXT@]. The memory cell which is read in step S1 is either memory cell @0, @1 or @2. This will be further explained hereinafter.

In step S2, the decoding processor selects input data from the memory cell @0, @1 or @2 which has been read in step S1. The input data is either the left-half portion of the memory cell or the right-half portion. The left-half portion is selected for input data when the

current bit is a binary zero (0): CB=0⇒LH@=ID. The right-half portion is selected for input data if the current bit is a binary one (1): CB=1⇒RH@=ID.

In step S3, the decoding processor extracts the address contained in the input data. This address defines the next-to-be-read memory cell: NXT@ = @∈ID.

In step S4, the decoding processor checks whether the input data contains a flag: √ ∈ ID ? If so, the decoding processor carries out steps S5 and S6. If not, the decoding processor will directly carry out step S1 anew for the subsequent bit in the data to be decoded.

In step S5, the decoding processor reads a memory cell which, in step S3, has been selected as the next-to-be-read memory cell: RD[NXT@]. This memory cell is either memory cell @10, @11, @12 or @13. It thus contains a symbol. The symbol is outputted.

In step S6, the decoding processor selects memory cell @0 as the next-to-be-read memory cell: NXT@ = @0. Subsequently, the decoding processor will carry out step S1 anew for the subsequent bit in the data to be decoded.

When the decoding processor carries out the series of steps S1-S6 for the first time, the current bit may not be the first bit of a code word. This does not matter. The effect will be that only the first decoded symbol is wrong. The symbols decoded subsequently will be correct. This auto-regulation feature is due to the fact that the data has been variable-length encoded in accordance with a Huffman tree.

The decoder illustrated in Fig. 3, which operates in accordance with the method illustrated in Fig. 4, is an example of an implementation of the basic features illustrated in Fig. 1. In this respect, the following remarks are made. The memory locations ML in Fig. 1 are implemented in Fig. 3 in the form of the right-half and left-half portions of the memory cells @1, @2 and @3. The output symbol data OSD in Fig. 1 are implemented in Fig. 3 in the form of addresses of the memory cells @10, @11, @12 and @13, which addresses are contained in the left-half portions of memory cells @0, @1, @2 and the right-half portion of the memory cell @2, respectively. The selection step SEL in Fig. 1 is implemented in Fig. 4 in the form of steps the S1-S3 and S6. In this respect, it is noted that the content of the most recently selected memory location in Fig. 4 is either the address of memory cell @1 or @2, or the flag √ which constitutes the end-of-code indication. In the latter case, the decoding processor will subsequently read memory cell @0, which represents the beginning of the Huffman tree. The evaluation step EVA in Fig. 1 is implemented in Fig. 4 in the form of the steps S4 and S5.

Fig. 5 illustrates an MPEG decoder in accordance with the invention. The MPEG decoder provides, in response to an MPEG data stream MDS, various types of decoded data, such as video data VID and audio data AUD, for example. The MPEG decoder

comprises an input section INP, a variable-length decoder VLD and further decoding circuitry FDC. The variable-length decoder VLD comprises a memory MEM and a decoding processor DPR. The memory MEM contains various Huffman trees HT[1]..HT[N] belonging to various types of data. The Huffman trees are stored in a manner as explained hereinbefore with reference to Fig. 1.

The MPEG decoder basically operates as follows. The input section INP recognizes the type of data which is currently being received. For example, the input section INP can recognize that data elements of the MPEG data stream represent a series of DCT coefficients or a motion vector, for example. The input section INP supplies an identification signal IS to the variable-length decoder VLD. The identification signal IS tells the variable-length decoder VLD, as it were, which type of data it currently receives. The decoding processor DPR uses the identification signal IS to read the correct Huffman tree HT from the memory MEM. More specifically, the identification signal IS determines the address at which the decoding processor DPR will start to read the memory for decoding a code word. Accordingly, the variable-length decoder VLD supplies variable-length decoded data, which may vary in type, to the further decoding circuitry FDC. The further decoding circuitry FDC further processes this data, which processing may include inverse quantization, discrete cosine transformation and motion compensation, for example.

The Huffman trees HT[1]..HT[N] contained in the memory MEM, as illustrated in Fig. 5, may have been pre-stored in a factory. However, one or more Huffman trees may also have been stored after the MPEG decoder left the factory. For example, a Huffman tree can be stored by means of memory-configuration data supplied to the MPEG decoder. The memory-configuration data will generally comprise the data to be stored in the memory MEM and information about where this data should be stored in the form of codes, for example. The memory configuration-data can be supplied to the MPEG decoder by means of a communication network like the Internet, for example.

The drawings and their description hereinbefore illustrate rather than limit the invention. It will be evident that there are numerous alternatives which fall within the scope of the appended claims. In this respect, the following closing remarks are made.

Any type of Huffman tree can be stored in a memory in a manner as illustrated in Fig. 1. The Huffman tree shown is this Figure is relatively simple. The Huffman tree shown in Fig. 3 is relatively simple too. The reason for this is to allow a good comprehension of the invention with a description that is concise. In practice, a Huffman tree will be more complicated. For example, the Huffman tree which is used in MPEG for coding DCT

coefficients is much more complicated. Nevertheless, this Huffman can be stored in a memory in accordance with the principle illustrated in Fig. 1. The Huffman trees shown in Figs. 1 and 3 are rather specific in that with each node there is one branch which has a leaf at its end, and another branch which is followed by further branches. This by no means excludes storage of Huffman trees which do not have this specific feature.

There are various manners to store a Huffman tree in a memory. Fig. 3 illustrates only one possible implementation in which a branch is represented by a left-half or a right-half portion of a memory cell. Another possible implementation is that each branch is represented by a different memory cell. In that case, a memory cell representing a branch which is followed by further branches, should contain a pointer pointing to the memory cells that represent these further branches. The value of the current bit from the data to be decoded will then determine which one of these memory cells should be read.

There are various manners to provide an output symbol. Fig. 3 illustrates only one possible implementation in which output symbols are stored in dedicated memory cells. A memory location which represents a branch having a leaf at its end, contains the address of the memory cell containing the appropriate output symbol. Another possible implementation is to directly store an output symbol in a memory location which represents a branch having a leaf at its end. An advantage of such an implementation is that it will require one step less to provide an output symbol compared with the possibility illustrated in Fig. 3.

There are numerous ways of implementing functions by means of items of hardware or software, or both. In this respect, the drawings are very diagrammatic, each representing only one possible embodiment of the invention. Thus, although a drawing shows different functions as different blocks, this by no means excludes that a single item of hardware or software carries out several functions. Nor does it exclude that a function is carried out by an assembly of items of hardware or software, or both.

For example, the decoding processor DPR illustrated in Fig. 3 can be implemented by means of a suitably programmed computer circuit. A set of instructions contained in a program memory can cause the computer circuit to effect various operations described hereinbefore with reference to Figs. 3 and 4. The set of instructions can be loaded into the program memory by reading a data carrier such as, for example, a disk that contains the set of instructions. The reading can be effected through a communication network such as, for example, the Internet. In that case, a service provider will make the set of instructions available to the public.

Any reference sign in a claim should not be construed as limiting the claim. The word "comprising" does not exclude the presence of other elements or steps than those listed in a claim. The word "a" or "an" preceding an element or step does not exclude the presence of a plurality of such elements or steps.

CLAIMS:

1. A decoder for decoding data which has been variable-length encoded in accordance with a Huffman tree (HT), characterized in that the decoder comprises:

- a memory (MEM) in which the Huffman tree is stored so that branches of the Huffman tree are represented by memory locations (ML); if a memory location represents a branch that is followed by subsequent branches, the memory locations contain a pointer (P) to memory locations representing these subsequent branches; if the memory location represents a branch that is not followed by subsequent branches, the memory location contains an end-of-code indication (ECI) and output-symbol data (OSD);
- a decoding processor (DPR) for carrying out the following steps with each data element (DE) in a sequence of data elements:
  - a selection step (SEL) in which a memory location is selected on the basis of the data element and the content of the most recently selected memory location; and
  - an evaluation step (EVA) in which it is checked whether the memory location that has been selected contains the end-of code indication and, if so, an output symbol (OS) is provided on the basis of the output-symbol data.

2. A method of decoding data which has been variable-length encoded in accordance with a Huffman tree (HT), characterized in that the method employs a memory in which the Huffman tree is stored so that branches (B) of the Huffman tree are represented by memory locations (ML); if a memory location represents a branch that is followed by subsequent branches, the memory location contains a pointer (P) to memory locations representing these subsequent branches; if the memory location represents a branch that is not followed by subsequent branches, the memory location contains an end-of-code indication (ECI) and output-symbol data (OSD),

the method being further characterized in that the following steps are carried out with each data element (DE) in a sequence of data elements:

- a selection step (SEL) in which a memory location is selected on the basis of the data element and the content of the most recently selected memory location; and
- a evaluation step (EVA) in which it is checked whether the memory location

that has been selected contains the end-of code indication and, if so, an output symbol (OS) is provided on the basis of the output-symbol data.

3. A method of configuring a decoder for decoding data which has been variable length encoded in accordance with a Huffman tree (HT), the method being characterized in that the Huffman tree (HT) is stored in a memory so that branches of the Huffman tree are represented by memory locations (ML); if a memory location represents a branch that is followed by subsequent branches, the memory location contains a pointer (P) to memory locations (ML) representing these subsequent branches; if the memory location represents a branch that is not followed by subsequent branches, the memory location contains an end-of-code indication (ECI) and output-symbol data (OSD) for generating an output symbol (OS).

4. A method of providing memory-configuration data enabling storage of a Huffman tree (HT) in a memory so that branches of the Huffman tree are represented by memory locations (ML); if a memory location represents a branch that is followed by subsequent branches, the memory location contains a pointer (P) to memory locations (ML) representing these subsequent branches; if the memory location represents a branch that is not followed by subsequent branches, the memory location contains an end-of-code indication (ECI) and output-symbol data (OSD) for generating an output symbol (OS).

5. A computer program product for a decoder comprising a memory (MEM) in which a Huffman tree is stored so that branches of the Huffman tree are represented by memory locations (ML); if a memory location represents a branch that is followed by subsequent branches, the memory locations contain a pointer (P) to memory locations representing these subsequent branches; if the memory location represents a branch that is not followed by subsequent branches, the memory location contains an end-of-code indication (ECI) and output-symbol data (OSD),
the computer program product comprising a set of instructions which, when loaded into the decoder, causes the decoder to effect the following steps with each data element (DE) in a sequence of data elements:

- a selection step (SEL) in which a memory location is selected on the basis of the data element and the content of the most recently selected memory location; and
- an evaluation step (EVA) in which it is checked whether the memory location that has been selected contains the end-of code indication and, if so, an output symbol (OS) is provided on the basis of the output-symbol data.

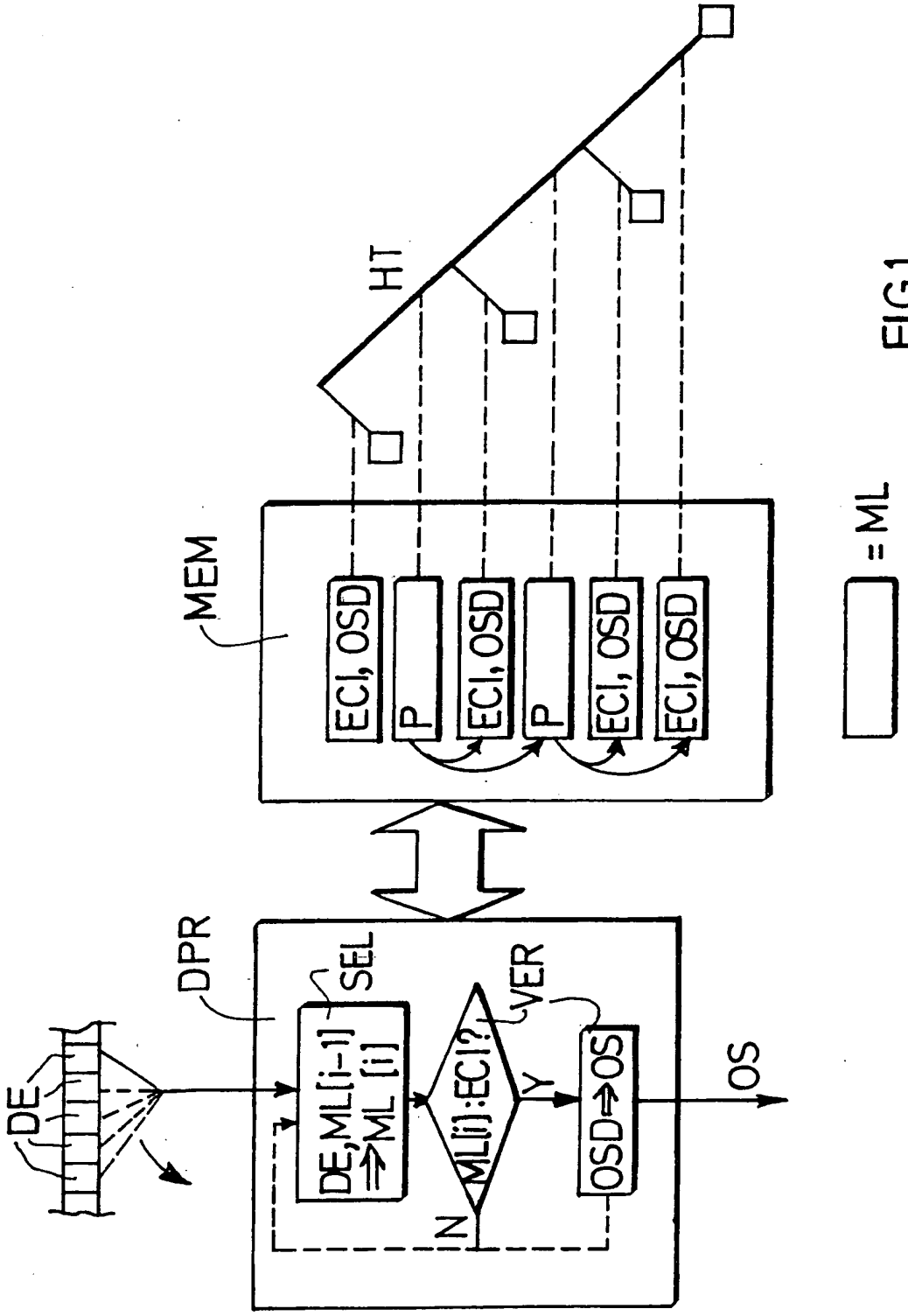

HT
MEM
ECI, OSD
P
ECI, OSD
P
ECI, OSD
ECI, OSD
= ML
DPR
SEL
VER
DE
DE,ML[i-1] ⇒ ML [i]
ML[i]:ECI?
Y
N
OSD⇒OS
OS

FIG.1

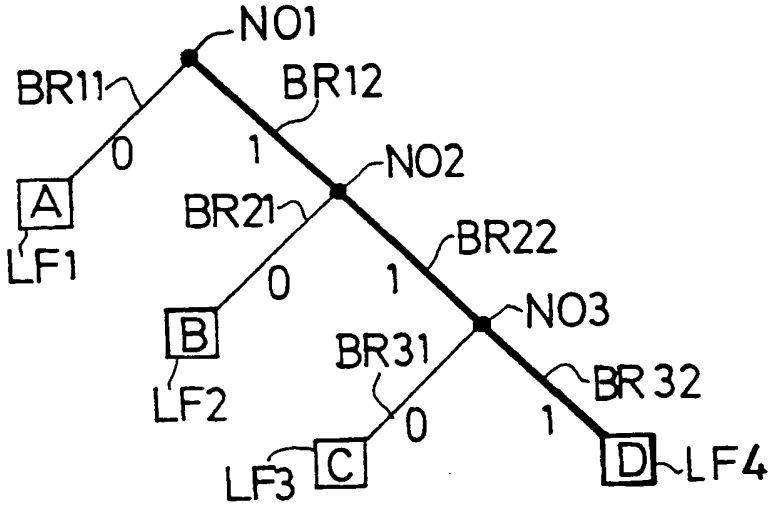

NO1
BR11
BR12
0
1
A
LF1
NO2
BR21
BR22
0
1
B
LF2
NO3
BR31
BR32
0
1
C
LF3
D
LF4

FIG.2

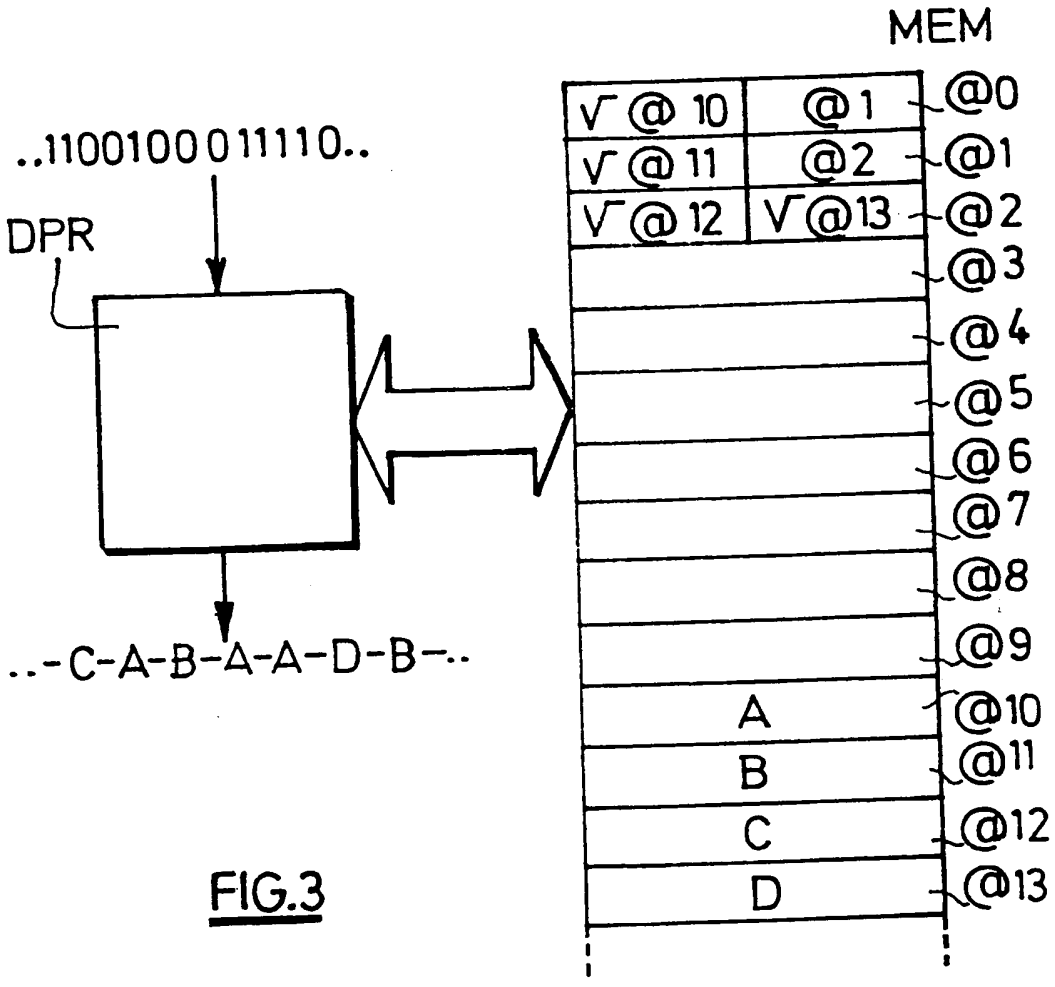

MEM
..1100100011110..
DPR
..-C-A-B-A-A-D-B-..
√@10
@1
@0
√@11
@2
@1
√@12
√@13
@2
@3
@4
@5
@6
@7
@8
@9
A
@10
B
@11
C
@12
D
@13

FIG.3

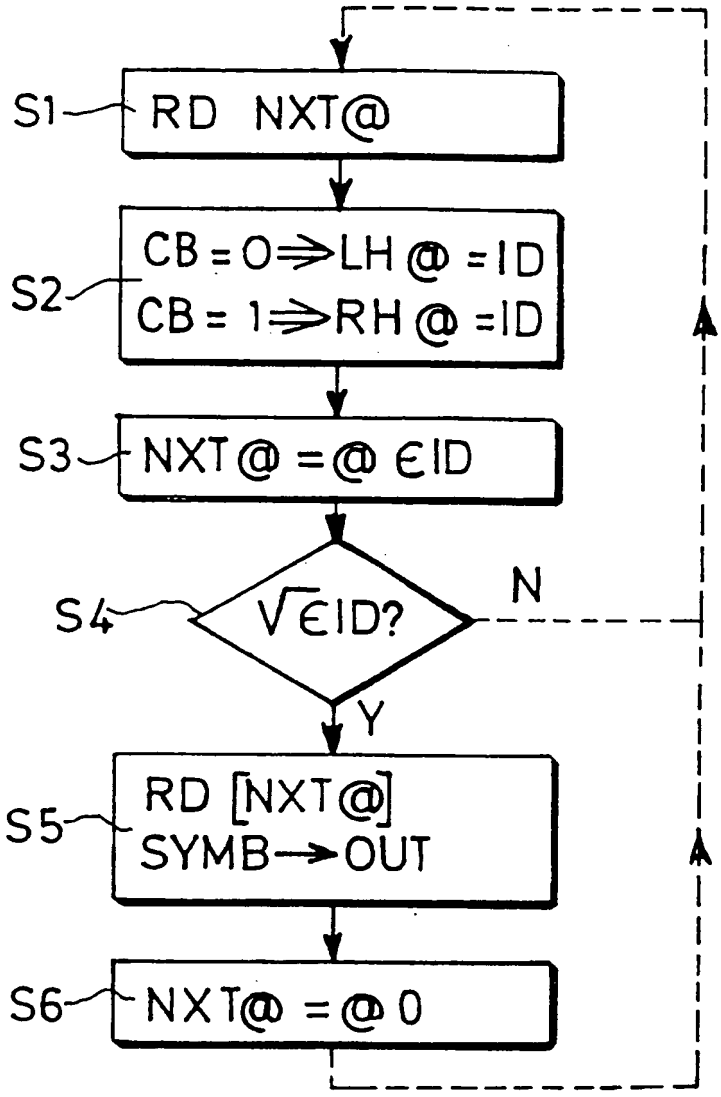

S1
RD NXT@
S2
CB = 0 ⇒ LH @ = ID
CB = 1 ⇒ RH @ = ID
S3
NXT @ = @ ∈ ID
S4
√ ∈ ID?
N
Y
S5
RD [NXT@]
SYMB → OUT
S6
NXT@ = @ 0

FIG.4

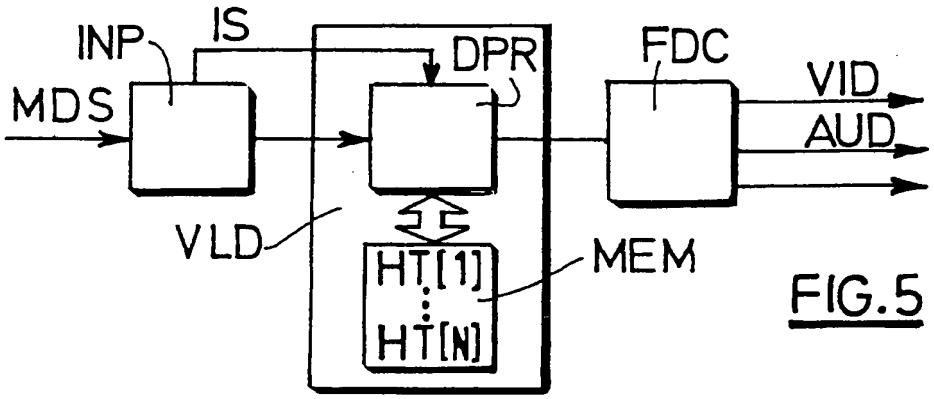

INP
IS
DPR
FDC
VID
AUD
MDS
VLD
HT[1]
HT[N]
MEM

FIG.5

INTERNATIONAL SEARCH REPORT

Inter onal Application No
PCT/EP 00/10715

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H03M7/42 H04N7/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H03M H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	AMAR MUKHERJEE ET AL: "MARVLE: A VLSI CHIP FOR DATA COMPRESSION USING TREE-BASED CODES" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS,US,IEEE INC. NEW YORK, vol. 1, no. 2, 1 June 1993 (1993-06-01), pages 203-214, XP000390613 ISSN: 1063-8210 abstract paragraph 'IV.B! ---	1-5
X	US 4 475 174 A (KANAYAMA HIDEAKI) 2 October 1984 (1984-10-02) abstract column 3, line 51 -column 4, line 49 --- -/--	1-5

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
5 January 2001	19/01/2001
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Berbain, F

Form PCT/ISA/210 (second sheet) (July 1992)